

Zdeněk Zastávka

Об алгоритмах с элементами памяти

Kybernetika, Vol. 4 (1968), No. 3, (201)--225

Persistent URL: <http://dml.cz/dmlcz/124633>

Terms of use:

© Institute of Information Theory and Automation AS CR, 1968

Institute of Mathematics of the Academy of Sciences of the Czech Republic provides access to digitized documents strictly for personal use. Each copy of any part of this document must contain these

Terms of use.



This paper has been digitized, optimized for electronic delivery and stamped with digital signature within the project *DML-CZ: The Czech Digital Mathematics Library*
<http://project.dml.cz>

Об алгоритмах с элементами памяти

Зденек Заставка

В статье определяется алгоритм с элементами памяти, заданный граф-схемой (это некоторое соответствие Л. А. Калужина [1]). Доказывается нормализуемость такого алгоритма (в смысле определения нормального алгоритма А. А. Маркова [2]).

ВВЕДЕНИЕ

Непосредственным импульсом для написания настоящей статьи является работа Л. А. Калужина [1]. После обоснования непригодности существующих до сих пор „классических“ теорий алгоритмов, а именно теории нормальных алгоритмов А. А. Маркова [2], для задач программирования для современных вычислительных машин,* Калужин предлагает новый вариант алгоритма — графические алгоритмы, граф-элем. Потом, кроме этого, автор обращает внимание на следующий факт: „Как самым рациональным образом ввести элементы памяти в граф-схему, причем так, чтобы сохранилась возможность и смысл операций подстановок, является важным, но пока не решенным вопросом“ ([2]; стр. 64).

В статье предлагается ввести памяти в такие граф-схемы, которые после интерпретации узлов можно определить как алгоритм композиции данных алгоритмов. Граф-схема названа *системой граф-схемой*, алгоритм — *композиционным алгоритмом*. Таким образом, принятая граф-схема будет отличаться от граф-схемы Калужина; соответствующие узлы будут интерпретироваться (цельными) алгоритмами, Калужин их интерпретирует (отдельными) операциями алгоритма (напр. у нормальных алгоритмов левыми частями формул подстановок и целыми этими формулами схемы, определяющей данный нормальный алгоритм).

Основные понятия определены в части I (глава 1 и 2). В теоремах 1—4 показано, что к каждой композиции алгоритмов существует композиционный алгоритм, который возникает путем такой интерпретации соответствующих узлов системной граф-схемы, что достаточно воспользоваться только компонированными алгоритмами (не надо нам прибавлять дальнейшие алгоритмы, как это наблюдается например при нормальных композициях), а также, что нам не надо расписывать алфавит, в котором даны эти алгоритмы. Мы получим таким образом элементарные виды композиционных алгоритмов.

* Калужин главным образом интересуется стандартными языками для алгоритмизации задач (именно математических и логических задач), которые пригодны к переводу на собственный язык конкретно данной вычислительной машины.

В части II проведено доказательство эквивалентности композиционных алгорифмов, определяемых только нормальными алгорифмами (т. н. квазинормальных композиционных алгорифмов) и нормальных алгорифмов.

Общее понятие алгорифма не более подробно определено; я ссылаюсь на цитированную работу Калужнина [1] и монографию А. А. Маркова [2], которая была исходным пунктом для моей работы. (Теория нормальных алгорифмов А. А. Маркова была началом и для Калужнина.) Поэтому все употребленные понятия теории алгорифмов, которые не будут подробнее определены, взяты из А. А. Маркова [2].

Я хотел бы выразить свою благодарность всем членам семинара на философском факультете Карлова университета во главе с д-ром философских наук проф. Отakarом Зихом за то, что я мог с ними консультировать вопросы касающиеся значения и проведения в жизнь этой работы. За очень конкретные замечания я благодарю профессора д-ра Иосифа Метелку и кандидата философских наук д-ра Павла Тихого, которые подробно подчитали первоначальный вариант этой работы. За обсуждение работы и важные замечания я признателен тоже Ивану Гавелу, пром. мат.

I. КОМПОЗИЦИОННЫЕ АЛГОРИФМЫ

1. Системные граф-схемы

Пусть дана некоторая плоскость и в ней

а) множество *окружностей* обозначенных через O_1, \dots, O_m ($m > 0$), лежащих вне себя и каждая вне другой; это множество обозначим через \mathbf{O} и назовем его множеством *операторов* O_1, \dots, O_m (следовательно $\mathbf{O} = \{O_1, \dots, O_m\}$);

б) множество *овалов* обозначенных через I_1, \dots, I_n ($n \geq 0$), лежащих вне себя и вне элементов множества \mathbf{O} и каждый из них лежит вне другого и вне каждого элемента множества \mathbf{O} ; это множество обозначим через \mathbf{I} и назовем его множеством *распознавателей* I_1, \dots, I_n (следовательно $\mathbf{I} = \{I_1, \dots, I_n\}$);

в) множество *прямоугольников* обозначенных через P_1, \dots, P_p ($p \geq 0$), лежащих вне себя и вне элементов множеств \mathbf{O} и \mathbf{I} и каждый из них лежит вне другого и вне элементов множеств \mathbf{O} и \mathbf{I} ; это множество обозначим через \mathbf{P} и назовем его множеством *памятей* P_1, \dots, P_p (следовательно $\mathbf{P} = \{P_1, \dots, P_p\}$);*

г) *точка* обозначена через V , которая не является элементом ни одного элемента множеств \mathbf{O} , \mathbf{I} и \mathbf{P} и находится вне этих элементов; эта точка — или множество состоящее из одного элемента, которое обозначим через \mathbf{V} (следовательно $\mathbf{V} = \{V\}$) — названа *входом*;

д) *точка* обозначена через Vy , различна от входа и не является элементом ни одного элемента множеств \mathbf{O} , \mathbf{I} и \mathbf{P} и находится вне этих элементов; эта точка — или множество состоящее из одного элемента, которое обозначим через \mathbf{Vy} (следовательно $\mathbf{Vy} = \{Vy\}$) — названа *выходом*.

* Для $m = 1$ или $n = 1$ или $p = 1$ будем пользоваться только обозначениями O , I , P ; следовательно $\mathbf{O} = \{O\}$, $\mathbf{I} = \{I\}$, $\mathbf{P} = \{P\}$.

Совокупность (систему) конечных не пересекающихся множеств V, O, I, P, Vy обозначим через \mathbf{M} :

$$\mathbf{M} = \{V, O, I, P, Vy\};$$

объединение системы множеств \mathbf{M} обозначим через $s\mathbf{M}$:

$$s\mathbf{M} = V \cup O \cup I \cup P \cup Vy.$$

Пусть m_1 и m_2 — два (равных или не равных) элемента объединения $s\mathbf{M}$; тогда скажем, что m_1 соединен направленным отрезком (стрелкой) с m_2 (или только: m_1 соединен с m_2), когда стрелка исходит из m_1 и кончается в m_2 или наоборот. (Графически это можно записать следующим образом: $\overrightarrow{m_1 m_2}$ или $\overleftarrow{m_1 m_2}$.)

У каждой стрелки *начало* (точка элемента объединения $s\mathbf{M}$, из которой стрелка исходит) и *конец* стрелки (точка элемента объединения $s\mathbf{M}$, в котором стрелка кончается).

Соединение которого-нибудь элемента объединения $s\mathbf{M}$ с которой-нибудь памятью, будет соединением этого элемента с какой-нибудь стороной данного прямоугольника — скажем, что это соединение является соединением этого элемента с какой-нибудь *стороной (данной) памяти*. (Стрелка из какой-нибудь памяти исходит или в ней кончается.) Если соединены все элементы объединения $s\mathbf{M}$ так, что исходя из какого-нибудь элемента, можно достигнуть любой другой, следуя по отрезкам (не обязательно всегда в направлении стрелки), то эту плоскую фигуру будем называть *системной граф-схемой* (с. г. с.), если соединение элементов соответствует следующим правилам:*

1. Во *входе* не кончается ни одна стрелка и исходит из него одна и только одна стрелка.
2. Из *выхода* не исходит ни одна стрелка.
3. В каждом *операторе* кончается хотя бы одна стрелка и исходит из него одна и только одна стрелка.
4. В каждом *распознавателе* кончается одна и только одна стрелка и исходят из него две и только две стрелки, отмеченные соответственно знаками + и — (*плюс-стрелка* и *минус-стрелка*), которые кончаются в какой-нибудь памяти.**

5.1. Каждую сторону памяти в которой кончается хотя бы одна стрелка, будем называть *входом памяти*. Концы всех кончающихся стрелок в каждом входе памяти *сливаются* (это значит, что все стрелки кончаются во входе памяти в одной и только в одной точке; см. рис. 1).

* Ср. [1], § 3, 1.

** Плюс- и минус-стрелка кончается каждая в одинаковой или различной стороне одной и той же памяти; из характера распознавателей следует, что это большей частью различные стороны.

В каждой памяти всегда хотя бы один вход памяти.

5.2. Каждую сторону памяти из которой исходит хотя бы одна стрелка, будем называть *выходом памяти*; выходом памяти может быть и вход памяти и наоборот. Начала всех исходящих стрелок в выходе памяти *не сливаются* (это значит, что стрелки исходят из выхода памяти из различных точек). Если исходит из выхода памяти n (> 1) стрелок, то стрелки нумерованы номерами $1, 2, \dots, n$ (*нумерованные стрелки**; см. рис. 2).

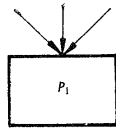


Рис. 1.

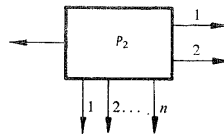


Рис. 2.

5.3. Противоположная сторона входа памяти есть всегда выходом памяти и наоборот. (Это значит, что сколько в памяти входов памяти, столько в ней и выходов памяти. В каждой памяти по крайней мере один вход памяти и один выход памяти и не больше четырех входов памяти и четырех выходов памяти.) В никакой стороне памяти не должно сливаться начало исходящей стрелки с концом никакой окончательной стрелки.

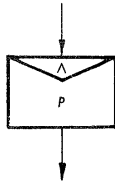


Рис. 3.

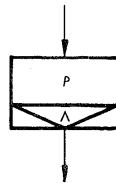


Рис. 4.

Вход памяти со специальным обозначением (рис. 3) мы будем называть *нон-входом памяти*; выход памяти со специальным обозначением (рис. 4) мы будем называть *нон-выходом памяти*.

В правилах 1—5.3 мы можем повсюду вместо входа памяти обдумывать нон-вход памяти и вместо выхода памяти нон-выход памяти.**

* Множество ($n > 1$) исходящих стрелок из каждого выхода памяти взаимно однозначно отображено на конечное множество $\{1, 2, \dots, n\}$ (единственную исходящую стрелку номером 1 не нумеруем!); это множество таким образом всегда упорядоченное множество.

** Мы не говорим, что если заменим в 5.1—5.3 „вход памяти“, „нон-входом памяти“, то мы должны заменить и „выход памяти“, „нон-выходом памяти“. Противоположной стороной входа памяти может быть нон-выход памяти, нон-входа памяти выход памяти или нон-выход памяти.

Таким образом соединенные элементы объединения sM мы будем называть узлами данной с. г. с.

2. Композиционный алгоритм

а) Дана с. г. с. с системой M , где $O = \{O_1, \dots, O_m\}$ ($m > 0$) и $I = \{I_1, \dots, I_n\}$ ($n \geq 0$).

б) Дан некоторый алфавит A .

в) Каждому узлу $O_i \in O$, где $1 \leq i \leq m$, (каждому оператору) соответствует алгоритм \mathfrak{D}_i в алфавите A (соответствие не нужно быть взаимно однозначным).

г) Каждому узлу $I_j \in I$, где $0 \leq j \leq n$, (каждому распознавателю) соответствует алгоритм \mathfrak{Z}_j в алфавите A (соответствие не нужно быть взаимно однозначным).

Условия а)–г) определяют *алгорифмическую интерпретацию* данной с. г. с., которую будем обозначать

$$\langle A; O \rightarrow \mathfrak{D}; I \rightarrow \mathfrak{Z} \rangle,$$

где $\mathfrak{D} = \{\mathfrak{D}_1, \dots, \mathfrak{D}_m\}$ и $\mathfrak{Z} = \{\mathfrak{Z}_1, \dots, \mathfrak{Z}_n\}$.*

Пусть дана некоторая алгорифмическая интерпретация

$$\langle A; O \rightarrow \mathfrak{D}; I \rightarrow \mathfrak{Z} \rangle.$$

Тогда всякую с. г. с. можно рассматривать как однозначное предписание для выполнения некоторого действия, в результате которого некоторые слова перейдут в новые слова в A . Как это действие мы осуществляем дано в следующем:

Пусть дана алгорифмическая интерпретация с. г. с. с системой M , где $P = \{P_1, \dots, P_p\}$ для $p > 0$. Всегда, когда слово S в алфавите A поступает в узел $P_k \in P$, где $1 \leq k \leq p$, этому узлу соответствует слово O , которое называется *содержанием памяти* P_k . Будем такое соответствие обозначать $P_k : O$.

Сформулируем теперь *правила образования содержания памяти*:

аа) Если в память P_k поступает слово S в алфавите A каким-нибудь из ее входов, то

(1) если это слово является *первым словом* поступающим в данную память, то ее содержанием будет слово S :

$$P_k : S : .$$

* Для $m = 1$ или $n = 1$ будем пользоваться только обозначениями $\mathfrak{D}, \mathfrak{Z}$; следовательно $\mathfrak{D} = \{\mathfrak{D}\}, \mathfrak{Z} = \{\mathfrak{Z}\}$.

- 206 (2) если перед поступлением этого слова было содержанием данной памяти слово O , то теперь содержанием памяти (после этого поступления слова S) является соединение слов O и S :

$$P_k : OS : .$$

(Если $S \neq \Lambda$, то можно сказать, что *содержание памяти расширилось*; если $S = \Lambda$, то *содержание памяти не тасирилось*.)

бб) Если в память P_k поступает слово S в алфавите A каким-нибудь из ее нон-входов, то

- (3) если это слово является *первым словом* поступающим в данную память, то ее содержанием будет *пустое слово*:

$$P_k : \Lambda : .$$

- (4) если это слово не является первым словом поступающим в данную память, то (существующее до сих пор) *содержание памяти не изменяется*. (Если было перед этим поступлением содержанием памяти слово O , то соответствие $P_k : O$: не изменяется.)

вв) Если из памяти P_k выходит слово (и пустое слово) по *последней стрелке* (по стрелке с наибольшим номером) какого-нибудь выхода памяти или нон-выхода памяти, содержанием памяти станет *пустое слово* (содержание памяти аннулировано):

$$P_k : \Lambda : .$$

После определения алгоритмической интерпретации $\langle A; \mathbf{O} \rightarrow \mathfrak{D}; I \rightarrow \mathfrak{I} \rangle$ данной с. г. с., использованием правил образования содержания памяти нам можно сформулировать правила преобразования поступающих слов в отдельные узлы с определением направления выхода (из этих узлов) преобразованных слов — *правила ориентированного преобразования слов*:

ааа) Если слово S в алфавите A поступает в узел V (на вход) — условимся это обозначать символом $V : S$: — то из этого узла продвигается по направлению выходящей стрелки в другой узел опять слово S .*

ббб) Если слово S в алфавите A поступает в узел $O_i \in \mathbf{O}$ (в оператор) и $\mathfrak{D}_i(S) = T$, где $\mathfrak{D}_i \in \mathfrak{D}$, то из этого узла продвигается по направлению выходящей стрелки к какой-нибудь (другой или тот же) узел слово T ; если не существует слово T , то из узла O_i не выходит ни одно слово.

ввв) Если слово S в алфавите A поступает в узел $I_j \in I$ (в распознаватель) и $\mathfrak{I}_j(S) = U$, где $\mathfrak{I}_j \in \mathfrak{I}$, то из этого узла продвигается в какой-нибудь (другой

* В каждой с. г. с. всегда хотя бы один оператор (см. а)), так что стрелка из входа никогда не кончается в выходе.

или тот же) узел слово U

- по направлению выходящей плюс-стрелки, когда $U = \Lambda$,
- по направлению выходящей минус-стрелки, когда $U \neq \Lambda$;

если не существует слово U , то из узла I_j не выходит ни одно слово.

гг) Пусть после поступления слова в алфавите A в узел $P_k \in P$ какой-нибудь его стороной

$$P_k : O : .$$

Тогда из этого узла продвигается по направлению выходящей стрелки из *противоположной стороны к данной стороне* с входом памяти или нон-выходом памяти в какой-нибудь (другой или тот же) узел слово S , где

- (1) $S = O$, если эта противоположная сторона *выход памяти*,
- (2) $S = \Lambda$, если эта противоположная сторона *нон-выход памяти*.

Если в этом выходе памяти или нон-выходе памяти нумерованные стрелки (стрелки с номерами $1, \dots, n; n > 1$), то слово S продвигается по направлению выходящей стрелки с номером $i + 1$, где

- (3) $i = 0$, если слово S является *первым словом* выходящим *этой стороной* или если *последним выходом* (перед выходом слова S) из этой стороны был выход какого-нибудь слова по *последней стрелке* (по стрелке с номером n);
- (4) i — номер стрелки, по которой выходило *последнее слово* (перед выходом слова S) из этой стороны, о котором знаем, что*

$$1 \leq i < n.$$

дд) Если слово S в алфавите A поступило в узел V_u (на выход) — условимся это обозначать символом $V_u : S$: — то из этого узла не выходит ни одно слово.
Алгоритмическая интерпретация

$$\langle A; O \rightarrow \mathfrak{D}; I \rightarrow \mathfrak{I} \rangle$$

и правила образования содержания памяти с правилами ориентированного преобразования слов (в данной с. г. с. с определенной системой множеств \mathbf{M}) определяют какой-нибудь алгоритм в алфавите A , который мы будем называть *композиционным алгоритмом* (к. а.) \mathfrak{R} в алфавите A . Слово S в алфавите A поступившее в узел V — *исходное слово* данного применения этого к. а.; если существует слово T в алфавите A поступившее в узел V_u , то мы будем его называть *результатом* данного применения этого к. а. к слову S :

$$\mathfrak{R}(S) = T.$$

* Выход слов выходом памяти или нон-выходом памяти с больше чем одной стрелкой (с нумерованными стрелками) осуществляется *постепенно* начиная первой стрелкой; после выхода последней стрелкой выходы слов повторяются начиная опять первой стрелкой.

3. Элементарные виды композиционных алгорифмов

Композиционные алгорифмы выгодны для композиции алгорифмов.* Покажем это на наиболее распространенных видах композиции алгорифмов: суперпозиции, объединении, повторении и разветвлении алгорифмов:

Теорема 1. *Каковы бы ни были алгорифмы $\mathfrak{D}_1, \dots, \mathfrak{D}_n$ ($n \geq 2$) в алфавите A , может быть построен такой к. а. \mathfrak{R} в алфавите A с алгорифмической интерпретацией $\langle A; \mathfrak{O} \rightarrow \mathfrak{D} \rangle$, что*

$$(1-1) \quad \mathfrak{R}(S) \simeq \mathfrak{D}_n(\mathfrak{D}_{n-1}(\dots(\mathfrak{D}_1(S)))) \quad (S - \text{слово в } A),$$

$$(1-2) \quad \mathfrak{O} = \{O_1, \dots, O_n\} \quad \text{и} \quad \mathfrak{D} = \{\mathfrak{D}_1, \dots, \mathfrak{D}_n\}.$$

Каждый такой к. а. \mathfrak{R} мы будем называть к. а. суперпозиции алгорифмов $\mathfrak{D}_1, \dots, \mathfrak{D}_n$.**

Доказательство теоремы 1 проведется построением такой с. г. с. с множеством операторов $\mathfrak{O} = \{O_1, \dots, O_n\}$ ($n \geq 0$), что после алгорифмической интерпретации $\langle A; \mathfrak{O} \rightarrow \mathfrak{D} \rangle$ (согласно условию (1-2)) получим к. а. \mathfrak{R} в алфавите A удовлетворяющий (1-1), как о том убедимся его применением к исходному слову S в алфавите A .***

С. г. с. имеет вид, изображенный на рис. 5.

Запись применения к. а. \mathfrak{R} к слову S проведен так, что кроме первой и последней строки с записанным поступлением слова S в узел V и поступлением результата применения этого алгорифма в узел V_y , на каждой строке записаны слова выходящие постепенно из отдельных узлов с. г. с.:

$V : S :$

$\mathfrak{D}_1(S)$

.....

$\mathfrak{D}_{n-1}(\dots(\mathfrak{D}_1(S)))$

$\mathfrak{D}_n(\mathfrak{D}_{n-1}(\dots(\mathfrak{D}_1(S))))$

$V_y : \mathfrak{D}_n(\mathfrak{D}_{n-1}(\dots(\mathfrak{D}_1(S)))) : .$

* Наш термин „композиция“ не соответствует этому же термину А. А. Маркова в [2]; „композиция алгорифмов“ в нашем значении для Маркова „сочетание алгорифмов“; „композиция алгорифмов“ Маркова — „суперпозиция алгорифмов“ для нас.

** Эта и следующие теоремы сформулированы согласно нормальным композициям А. А. Маркова в [2], IV.

*** Приведенная с. г. с. не имеет ни одной памяти; потому здесь не приходят во внимание правила образования содержания памяти.

Слово поступающее в узел V_y , т. е. результат применения к. а. \mathfrak{R} в алфавите A к исходному слову S , есть слово $\mathfrak{D}_n(\mathfrak{D}_{n-1}(\dots(\mathfrak{D}_1(\epsilon))))$. Этот результат хватит к подтверждению действительности условия (1-1):

$$\mathfrak{R}(S) \simeq \mathfrak{D}_n(\mathfrak{D}_{n-1}(\dots(\mathfrak{D}_1(S)))) .$$



Рис. 5.

Замечание. Доказательство теоремы 1 приведено значительно сокращенным образом. Существенную часть совершенного доказательства помимо с. г. с. здесь замещает запись применения к. а. \mathfrak{R} к исходному слову. Совершенное доказательство зависело бы от приведения доказательства следующих лемм:

Лемма 1.1. Если имеет смысл выражение $\mathfrak{D}_n(\mathfrak{D}_{n-1}(\dots(\mathfrak{D}_1(S))))$, то к. а. \mathfrak{R} применим к слову S и

$$\mathfrak{R}(S) = \mathfrak{D}_n(\mathfrak{D}_{n-1}(\dots(\mathfrak{D}_1(S)))) .$$

Лемма 1.2 Если к. а. \mathfrak{R} применим к слову S в алфавите A , то имеет смысл выражение $\mathfrak{D}_n(\mathfrak{D}_{n-1}(\dots(\mathfrak{D}_1(S))))$.

Так как речь идет о аналогии доказательств теорем о нормальных алгоритмах А. А. Маркова (см. [2], III) и так как наши теоремы и доказательства являются значительно более простые, мы не считаем необходимым и целесообразным (именно ввиду обширности этого доказательства) здесь целое доказательство выполнить.

Это аналогично действительно и для в дальнейшем приведенные теоремы и их доказательства.

Теорема 2. *Каковы бы ни были алгоритмы $\mathfrak{D}_1, \dots, \mathfrak{D}_n$ ($n \geq 2$) в алфавите A , может быть построен такой к. а. \mathfrak{R} в алфавите A с алгоритмической интер-*

210 претацией $\langle A; \mathbf{O} \rightarrow \mathfrak{D} \rangle$, что

$$(2-1) \quad \mathfrak{R}(S) \simeq \mathfrak{D}_1(S) \dots \mathfrak{D}_n(S) \quad (S - \text{слово в } A),$$

$$(2-2) \quad \mathfrak{D} = \{O_1, \dots, O_n\} \quad \text{и} \quad \mathfrak{D} = \{\mathfrak{D}_1, \dots, \mathfrak{D}_n\}.$$

Каждый такой к. а. \mathfrak{R} мы будем называть к. а. объединения алгоритмов $\mathfrak{D}_1, \dots, \mathfrak{D}_n$.

С. г. с. для этого доказательства имеет кроме множества \mathbf{O} и множество $\mathbf{P} = \{P_1, P_2\}$ (рис. 6).

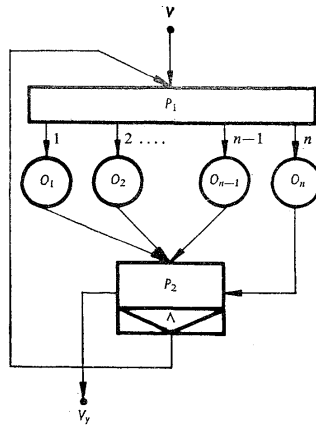


Рис. 6.

Запись применения этого к. а. \mathfrak{R} к слову S имеет кроме строк с исходным словом, выходящими словами из операторов (узлов O_1, \dots, O_n) и словом поступающим на выход, как этому было в доказательстве теоремы 1, также строки с содержанием отдельных памятей (узлов P_1 и P_2) и у которых обозначено стрелком (или нумерованной стрелкой) направление дальнейшего продвижения слов выходящих из этих узлов (согласно правилам образования содержания памятей и соответствующим правилам ориентированного преобразования слов); направление данное стрелкой исходящей из нон-выхода памяти имеет символ Λ (это предупреждение о выходящем из этого узла слове Λ); после выхода слова последней нумерованной стрелкой обозначено содержание памяти (это пустое слово — ануллирование содержания памяти) в квадратных скобках:

$V : S :$

$P_1 : S : \downarrow 1$

$\mathfrak{D}_1(S)$
 $P_2 : \mathfrak{D}_1(S) : \downarrow \wedge$
 $P_1 : S : \downarrow 2$
 $\mathfrak{D}_2(S)$
 $P_2 : \mathfrak{D}_1(S) \mathfrak{D}_2(S) : \downarrow \wedge$

 $P_2 : \mathfrak{D}_1(S) \mathfrak{D}_2(S) \dots \mathfrak{D}_{n-1}(S) : \downarrow \wedge$
 $P_1 : S : \downarrow n$
 $[P_1 : \wedge :]$
 $\mathfrak{D}_n(S)$
 $P_2 : \mathfrak{D}_1(S) \mathfrak{D}_2(S) \dots \mathfrak{D}_{n-1}(S) \mathfrak{D}_n(S) : \leftarrow$
 $\forall \gamma : \mathfrak{D}_1(S) \mathfrak{D}_2(S) \dots \mathfrak{D}_{n-1}(S) \mathfrak{D}_n(S) : .$

Теорема 3. *Каковы бы ни были алгоритмы \mathfrak{D} и \mathfrak{Z} в алфавите A , может быть построен такой к. а. \mathfrak{R} в алфавите A с алгоритмической интерпретацией $\langle A; \mathfrak{O} \rightarrow \mathfrak{D}; I \rightarrow \mathfrak{Z} \rangle$, что*

$$(3-1) \quad \mathfrak{R}(S) = T \quad (S - \text{слово в } A),$$

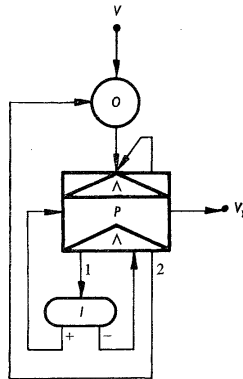


Рис. 7.

тогда и только тогда, когда существует ряд слов S_0, S_1, \dots, S_n ($n > 0$) удовлетворяющий условиям

- (i) $S_0 = S,$
- (ii) $S_i = \mathfrak{D}(S_{i-1}) \quad (0 < i \leq n),$

- (iii) $S_n = T,$
- (iv) $\mathfrak{Z}(S_i) \neq \Lambda \quad (0 < i < n),$
- (v) $\mathfrak{Z}(S_n) = \Lambda,$
- (3-2) $O = \{O\}, \quad I = \{I\} \quad \text{и} \quad \mathfrak{D} = \{\mathfrak{D}\}, \quad \mathfrak{Z} = \{\mathfrak{Z}\}.$

Каждый такой к. а. \mathfrak{R} мы будем называть к. а. повторения алгоритма \mathfrak{D} , управляемое алгоритмом \mathfrak{Z} .

С. г. с. имеет вид, изображенный на рис. 7 ($O = \{O\}, I = \{I\}, P = \{P\}$).

Запись применения к. а. \mathfrak{R} к исходному слову S в алфавите A :

$V : S :$
 $\mathfrak{D}(S)$
 $P : \mathfrak{D}(S) : \downarrow 1$
 $\mathfrak{Z}(\mathfrak{D}(\epsilon) \downarrow + * \quad \mathfrak{Z}(\mathfrak{D}(S)) \downarrow - *$
 $P : \mathfrak{D}(S) : \rightarrow ** \quad P : \mathfrak{D}(S) : \uparrow \wedge **$
 $\forall \gamma : \mathfrak{D}(S) : \quad P : \mathfrak{D}(S) : \downarrow 2$
 $[P : \wedge :]$
 $\mathfrak{D}(\mathfrak{D}(S))$
 $P : \mathfrak{D}(\mathfrak{D}(S)) : \downarrow 1$
 $\mathfrak{Z}(\mathfrak{D}(\mathfrak{D}(S))) \downarrow + \quad \mathfrak{Z}(\mathfrak{D}(\mathfrak{D}(S))) \downarrow -$
 $P : \mathfrak{D}(\mathfrak{D}(S)) : \rightarrow \quad P : \mathfrak{D}(\mathfrak{D}(S)) : \uparrow \wedge$
 $\forall \gamma : \mathfrak{D}(\mathfrak{D}(S)) : \quad P : \mathfrak{D}(\mathfrak{D}(S)) : \downarrow 2$
 $[P : \wedge :]$
 $\dots\dots\dots$
 $\forall \gamma : \mathfrak{D}(\mathfrak{D}(\dots(\mathfrak{D}(S)))) : .$

Теорема 4. Каковы бы ни были алгоритмы $\mathfrak{D}_1, \mathfrak{D}_2, \mathfrak{Z}$ в алфавите A , может быть построен такой к. а. \mathfrak{R} в алфавите A с алгоритмической интерпретацией $\langle A; \mathfrak{D} \rightarrow \mathfrak{D}; I \rightarrow \mathfrak{Z} \rangle$, что

- (4-1.1) $\mathfrak{R}(S) \simeq \mathfrak{D}_1(S) \quad (S - \text{слово в } A, \mathfrak{Z}(S) = \Lambda),$
- (4-1.2) $\mathfrak{R}(S) \simeq \mathfrak{D}_2(S) \quad (S - \text{слово в } A, \mathfrak{Z}(S) \neq \Lambda),$
- (4-2) $O = \{O_1, O_2\}, \quad I = \{I\} \quad \text{и} \quad \mathfrak{D} = \{\mathfrak{D}_1, \mathfrak{D}_2\}, \quad \mathfrak{Z} = \{\mathfrak{Z}\}$

* Таким образом обозначен выход слова $\mathfrak{Z}_j(S')$ (S' — слово поступающее в узел I_j ; здесь $I_j = I, S' = \mathfrak{D}(S)$) плюс-стрелкой и минус-стрелкой.

** Согласно правилам ориентированного преобразования слов известно, что из плюс-стрелки выходит всегда слово \wedge .

и что \mathfrak{A} применим к тем и только к тем словам в алфавите A , к которым применим \mathfrak{I} . Каждый такой к. а. \mathfrak{A} мы будем называть к. а. разветвления алгоритмов \mathfrak{D}_1 , и \mathfrak{D}_2 , управляемое алгоритмом \mathfrak{I} .

С. г. с. имеет вид, изображенный на рис. 8. ($I = \{I\}$, $P = \{P\}$).

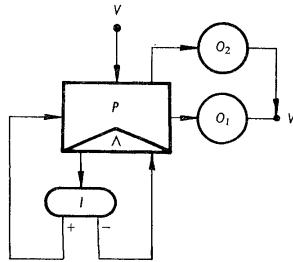


Рис. 8.

Запись применения к. а. \mathfrak{A} к исходному слову S в алфавите A :

$V : S :$
 $P : S : \downarrow$
 $\mathfrak{I}(S) \downarrow + \quad \mathfrak{I}(S) \downarrow -$
 $P : S : \rightarrow \quad P : S : \uparrow$
 $\mathfrak{D}_1(S) \quad \mathfrak{D}_2(S)$
 $Vy : \mathfrak{D}_1(S) : \quad Vy : \mathfrak{D}_2(S) : .$

II. КВАЗИНОРМАЛЬНЫЕ КОМПОЗИЦИОННЫЕ АЛГОРИФМЫ И НОРМАЛЬНЫЕ АЛГОРИФМЫ

1. Квазинормальные композиционные алгоритмы

Алгоритмическую интерпретацию

$$\langle A; O \rightarrow \mathfrak{D}; I \rightarrow \mathfrak{I} \rangle$$

некоторой с. г. с. мы будем называть *нормальной алгоритмической интерпретацией*, если алгоритмы \mathfrak{D}_i ($i \in \mathfrak{D}$) и \mathfrak{I}_j ($j \in \mathfrak{I}$) нормальные алгоритмы в алфавите A .

Каждый к. а. заданный нормальной алгоритмической интерпретацией мы будем называть *квазинормальным композиционным алгоритмом* (к. к. а.).

2. Нормальный алгоритм эквивалентен квазинормальному алгоритму

Ясно, что имеет место

Теорема 5. Для любого нормального алгоритма \mathfrak{A} в алфавите A существует такой к. к. а. \mathfrak{A} в алфавите A , что

$$\mathfrak{A}(S) \simeq \mathfrak{A}(S) \quad (S - \text{слово в } A).$$

Докажем, что имеет место

Теорема 6. Для любого к. к. а. \mathfrak{A} в алфавите A существует такой нормальный алгоритм \mathfrak{A} над алфавитом A , что

$$\mathfrak{A}(S) \simeq \mathfrak{A}(S) \quad (S - \text{слово в } A).$$

Доразательство этой теоремы будем вести подобным способом, как в работе Н. П. Тер-Захаряна [4] доказываемость эквивалентности квазинормальных и нормальных алгоритмов (где квазинормальными алгоритмами названы алгоритмы получаемые нормальной интерпретацией граф-схем в смысле Л. А. Калужнина). Пусть

$$\langle A; \mathbf{0} \rightarrow \mathfrak{D}; I \rightarrow \mathfrak{S} \rangle$$

нормальна алгоритмическая интерпретация данной с. г. с. на которой был определен к. к. а. \mathfrak{A} в алфавите A .

Концы окончательных стрелок в входах памяти или в нон-входах всех памятей (узлов P_1, \dots, P_p) мы будем называть квази-узлами (*к-узлами*) данной с. г. с. Каждая память (узел $P_k \in \mathbf{P}$, для $1 \leq k \leq p$) так с одной до четырех к-узлов.

Противоположностью входа или оператора (узла V или узла $O_i \in \mathbf{O}$, где $1 \leq i \leq m$) мы будем называть узел (если речь идет о операторе, распознавателе или выходе) или к-узел (если речь идет о к-узле какой-то памяти), к которому подходит стрелка из входа или данного оператора.

Плюс и минус-противоположностью распознавателя мы будем называть узел (речь идет о операторе, другом распознавателе или выходе) к которому подходит плюс и минус-стрелка из данного распознавателя.

Противоположностью или *нон-противоположностью* к-узла данного входа памяти или нон-входа памяти мы будем называть узел (если речь идет о операторе, распознавателе или выходе) или к-узел (если речь идет о к-узле некоторой памяти), к которому подходит стрелка (не нумерованная стрелка) из противоположного выхода памяти или нон-выхода памяти.

Упорядоченными противоположностями или *упорядоченными нон-противоположностями* к-узла данного входа памяти или нон-входа памяти мы будем называть все узлы (если речь идет о операторах, распознавателях или выходе) или к-узлы (если речь идет о к-узлах некоторых памятей), к которым подходят

нумерованные стрелки из *противоположного выхода памяти или не-выхода памяти*; если стрелки имеют номера $1, \dots, n$, потом мы скажем, что k -узел имеет *первую по n -той противоположности или не-противоположности*.

Знаем, что m и n числа операторов и распознавателей; пусть r — число k -узлов и $s = r + m + n$. Если выходу поставим в соответствие номер $s + 1$ и каждому оператору, распознавателю и k -узлу всегда только один из номеров $1, \dots, s$, то

- множество номеров поставленных в соответствие операторам (элементам множества \mathbf{O}) мы будем обозначать через \mathbf{On} ;
- множество номеров поставленных в соответствие распознавателям (элементам множества \mathbf{I}) мы будем обозначать через \mathbf{In} ;
- множество номеров поставленных в соответствие k -узлам в входах памяти с противоположностью мы будем обозначать через \mathbf{Pn} ;
- множество номеров поставленных в соответствие k -узлам в не-входах памяти с противоположностью мы будем обозначать через \mathbf{Pn}^\wedge ;
- множество номеров поставленных в соответствие k -узлам в входах памяти с не-противоположностью мы будем обозначать через \mathbf{Pn}_\wedge ;
- множество номеров поставленных в соответствие k -узлам в не-входах памяти с не-противоположностью мы будем обозначать через $\mathbf{Pn}_\wedge^\wedge$;
- множество номеров поставленных в соответствие k -узлам в входах памяти с упорядоченными противоположностями мы будем обозначать через \mathbf{Pno} ;
- множество номеров поставленных в соответствие k -узлам в не-входах памяти с упорядоченными противоположностями мы будем обозначать через \mathbf{Pno}^\wedge ;
- множество номеров поставленных в соответствие k -узлам в входах памяти с упорядоченными не-противоположностями мы будем обозначать через \mathbf{Pno}_\wedge ;
- множество номеров поставленных в соответствие k -узлам в не-входах памяти с упорядоченными не-противоположностями мы будем обозначать через $\mathbf{Pno}_\wedge^\wedge$.

Соединение множеств \mathbf{Pn} , \mathbf{Pn}^\wedge , \mathbf{Pn}_\wedge и $\mathbf{Pn}_\wedge^\wedge$ мы будем обозначать через \mathbf{Pn} :

$$\mathbf{Pn} = \mathbf{Pn} \cup \mathbf{Pn}^\wedge \cup \mathbf{Pn}_\wedge \cup \mathbf{Pn}_\wedge^\wedge;$$

аналогично

$$\mathbf{Pno} = \mathbf{Pno} \cup \mathbf{Pno}^\wedge \cup \mathbf{Pno}_\wedge \cup \mathbf{Pno}_\wedge^\wedge.$$

Для $1 \leq i \leq s$ пусть $\Theta(i)$ — число противоположностей или не-противоположностей или упорядоченных противоположностей или упорядоченных не-противоположностей или плюс или минус-противоположностей узла или k -узла с поставленным в соответствие номером i ; ясно, что $\Theta(i) = 1$, если $i \in \mathbf{On} \cup \mathbf{Pn}$; $\Theta(i) = 2$, если $i \in \mathbf{In}$; $\Theta(i) > 1$, если $i \in \mathbf{Pno}$.

Определим функцию $\Psi(i, j)$, для $1 \leq i \leq s$, $1 \leq j \leq \Theta(i)$:

$$\Psi(i, j) = \begin{cases} \text{номер противоположности или non-противоположности узла} \\ \text{или к-узла с номером } i, \text{ если } i \in \mathbf{On} \cup \mathbf{Pn} \text{ (всегда } j = 1); \\ \text{номер плюс-противоположности узла с номером } i, \text{ если} \\ i \in \mathbf{ln} \text{ и } j = 1; \\ \text{номер минус-противоположности узла с номером } i, \text{ если} \\ i \in \mathbf{ln} \text{ и } j = 2; \\ \text{номер } j\text{-ой противоположности или non-противоположности} \\ \text{к-узла с номером } i, \text{ если } i \in \mathbf{Pno}. \end{cases}$$

Искомый нормальный алгоритм \mathfrak{A} над алфавитом A построим как суперпозицию трех алгоритмов:

$$\mathfrak{A} = \mathfrak{D} \circ \mathfrak{C} \circ \mathfrak{B},$$

где \mathfrak{B} будет такой алгоритм, который к входному слову S „присоединит“ со специальным обозначением эти основные информации о с. г. с. (о нумерации узлов и их соединении):

- во первых здесь будут введены (в специальном устройении) элементы множества \mathbf{Pno} ;
- далее потом алгоритмы, которые будут „выражать“ способ трансформации слов в отдельных узлах и к-узлах (это будет по существу устроенная алгоритмическая переписка правил 1—5.3);
- наконец здесь будет отметкой „ Δ “ выражено до которого узла поступающее слово поступает.

Алгоритм \mathfrak{C} будет „моделировать“ настоящее применение к. к. а. \mathfrak{A} к исходному слову; он будет построен как повторение суперпозиции нормальных алгоритмов \mathfrak{E} и \mathfrak{F} , управляемое нормальным алгоритмом \mathfrak{B} , который будет „устанавливать“, если полученное слово („междурезультат“) уже результатом применения алгоритма \mathfrak{A} . Направление поступления „междурезультата“ на следующий узел в с. г. с. будет назначено при помощи введенной отметки „ Δ “ (отметка будет всегда перед „моделирующим“ алгоритмом данного узла или к-узла с поставленным в соответствие номером). Результат трансформации поступленного слова на соответствующий узел будет „обеспечивать“ алгоритм \mathfrak{F} и алгоритм \mathfrak{E} потом „проведет“ передвижение отметки „ Δ “ согласно соединению этого узла со следующим узлом, куда трансформированное слово будет поступать.

Алгоритм \mathfrak{D} будет принят к слову, которому надо поступить на выход — он нам обеспечит, что слово, которое поступит на выход, будет в самом деле словом $\mathfrak{A}(S)$.

Замечание. К характеристике основной идеи доказательства, когда „междурезультатом“ всегда какое-то слово, к которому присоединена „легенда“ с записью откуда слово „прихо-

дило⁴ и куда ему поступить, позволяю себе привести это сравнение, которое сформулировал профессор д-р Иосиф Метелка, когда прочитал первую версию этой статьи: „Может быть, что отцом этой идеи была снова практика, именно практика применяемая обычно во время производства частей, которые проходят многими производственными операциями. Эти части с собой в течении всего производственного процесса несут сопроводительную карточку, обозначающую порядок работ. При каждом процессе обрабатается не только продукт, но и сопроводительная карточка, именно иногда в зависимости от самого продукта, если например только что исполнимой операцией был контроль, который установил какой-то недостаток.“

2. 1. Алгоритм \mathfrak{B}

Операторам, распознавателям и к-узлам в данной с. г. с. мы поставим в соответствие номера $1, \dots, s$; к каждому номеру $1 \leq i \leq s$ определим алгоритм \mathfrak{A}_i , который будет „алгоритмическим выражением“ функции данного узла или к-узла с. г. с.

Каждому номеру $i \in \mathbf{Pno}$ поставим в соответствие знак (букву) $[i]_0$ и из всех этих букв составим алфавит $[.]_0$. Каждому номеру $i \in \mathbf{On} \cup \mathbf{In} \cup \mathbf{Pn} \cup \mathbf{Pno}$ поставим в соответствие знак (букву) $[i]_1$ и из всех этих букв составим алфавит $[.]_1$.

Символом U^{L-1} обозначим любое слово удовлетворяющее следующим условиям:

1. в слово U^{L-1} входят все буквы алфавита $[.]_0$;
2. пусть $i \in \mathbf{Pno}$, и слово

$$\underbrace{[i]_0 \dots [i]_0}_a \text{ раз}$$

обозначено символом $[i]_0^a$; * тогда для каждое i в слово U^{L-1} входит слово $[i]_0^a$ всегда только один раз и всегда $1 \leq a \leq \Theta(i) + 1$.

Когда нас будет интересовать в слове U^{L-1} (первое и единственное) вхождение слова $[i]_0^a$ с данным i , потом слово

$$U_1 [i]_0^a U_2,$$

где U_1 есть начало слова U^{L-1} и U_2 есть конец слова U^{L-1} , мы будем обозначать символом U^{L-1} , так что

$$U^{L-1} = U_1 [i]_0^a U_2.$$

Замечание. Как увидим далее в тексте, вхождение слова $[i]_0^a$ в слово U^{L-1} для фиксированного номера $i \in \mathbf{Pno}$, нас будет интересовать много раз; это вхождение точно записано имеет этот вид (см.: А. А. Марков [2], I, § 4):

$$U_1 * [i]_0^a * U_2,$$

* Подобным образом (в следующем будем употреблять)

$$[i]_1^a = \underbrace{[i]_1 \dots [i]_1}_a \text{ раз}$$

где U_1 левым крылом и U_2 правым крылом этого вхождения. Таким образом данное вхождение нам даст возможность каждое слово $U^{[1]}$ написать в виде

$$U_1 \bar{U} \bar{U}_2$$

а именно для так написанного слова $U^{[1]}$ мы выбрали обозначение $U^{[1]}$.

Пусть все нужные информации о с. г. с. выражены в слове S^B :

$$S^B = U^{[1]} \bar{\delta} [1]_1 \mathfrak{A}_1^n \dots [v-1]_1 \mathfrak{A}_{v-1}^n \Delta [v]_1 \mathfrak{A}_v^n [v+1]_1 \mathfrak{A}_{v+1}^n \dots \mathfrak{A}_s^n [s+1]_1,$$

где v — номер противоположности входа и для каждого слова $[i]_1^n$ входящего в слово $U^{[1]}$ есть $a = 1$; $\bar{\delta}$ — „отделяющий“ знак.

Объединением алфавитов, в которых запишем изображение алгоритмов $\mathfrak{A}_1, \dots, \mathfrak{A}_s$ (т. е. алгоритмов $\mathfrak{A}_1^n, \dots, \mathfrak{A}_s^n$), будет алфавит \mathcal{C} .*

Алгоритм \mathfrak{B} построим теперь в алфавите $\mathcal{C} \cup [.]_1 \cup \{\delta, \omega, \Delta\}$ со схемой

$$\begin{cases} \omega \xi \rightarrow \xi \omega (\xi \in A) \\ \omega \rightarrow \cdot S^B \\ \rightarrow \omega \end{cases}$$

Пусть S — исходное слово в алфавите A , тогда

$$\mathfrak{B}(S) = SS^B.$$

2.2. Алгоритм \mathcal{C}

Первым исходным словом алгоритма \mathcal{C} является таким образом слово SS^B , которое особым случаем слова

$$PU^{[1]} \bar{\delta} [1]_1 \mathfrak{A}_1^n \dots [i-1]_1 \mathfrak{A}_{i-1}^n \Delta [i]_1 \mathfrak{A}_i^n [i+1]_1 \mathfrak{A}_{i+1}^n \dots [s]_1 \mathfrak{A}_s^n [s+1]_1,$$

где P — слово в A .

Дальнейшие исходные слова этого алгоритма будут содержать слова, которые будут содержаниями отдельных памятей. К каждому узлу $P_k \in \mathbf{P}$ (в каждой памяти) построим поэтому алфавит двойников букв алфавита A , который обозначим через \bar{A}^k ; объединение всех этих алфавитов будет алфавит \bar{A} :

$$\bar{A} = \bar{A}^1 \cup \dots \cup \bar{A}^p.$$

Пусть таким образом является исходным словом алгоритма \mathcal{C} слово

$$PU^{[1]} \bar{Q} \bar{\delta} [1]_1 \mathfrak{A}_1^n \dots [i-1]_1 \mathfrak{A}_{i-1}^n \Delta [i]_1 \mathfrak{A}_i^n [i+1]_1 \mathfrak{A}_{i+1}^n \dots [s]_1 \mathfrak{A}_s^n [s+1]_1,$$

где \bar{Q} — слово в \bar{A} .

* Надо нам осознать, что $A \cup \bar{A} \cup [.]_0 \cup \{\blacktriangle\} \subset \mathcal{C}$. Принимая во внимание определения дальнейших алгоритмов, не должен алфавит \mathcal{C} строиться как расширение объединения

$$[.]_1 \cup [.]_2 \cup \{[1]_3, \dots, [s+1]_3\} \cup \{[1]_4, \dots, [s+1]_4\} \cup \{\delta, \omega, \Delta\}.$$

2.2.1. Алгоритм \mathfrak{E}

Для

$$\mathfrak{E} = (\mathfrak{E} \circ \mathfrak{F}) | \mathfrak{E}$$

алгоритм \mathfrak{F} построим так, чтобы

$$\begin{aligned} & \mathfrak{F}(PU^{l-1}\bar{Q}\delta[1]_1 \mathfrak{A}_1^m \dots [i-1]_1 \mathfrak{A}_{i-1}^m \triangle [i]_1 \mathfrak{A}_i^m [i+1]_1 \mathfrak{A}_{i+1}^m \dots [s]_1 \mathfrak{A}_s^m [s+1]_1) = \\ & = \mathfrak{A}_i(PU^{l-1}\bar{Q}) \delta[1]_1 \mathfrak{A}_1^m \dots [i-1]_1 \mathfrak{A}_{i-1}^m \triangle \mathfrak{A}_i^m [i+1]_1 \mathfrak{A}_{i+1}^m \dots [s]_1 \mathfrak{A}_s^m [s+1]_1. \end{aligned}$$

Результат действия этого алгоритма будет зависеть от узла или k -узла s поставимым в соответствие номером i . Построим теперь алгоритмы $\mathfrak{A}_1, \dots, \mathfrak{A}_s$ такие, что

– если $i \in \mathbf{On}$, то

$$\mathfrak{A}_i(PU^{l-1}\bar{Q}) = \mathfrak{D}_i(P) U^{l-1}\bar{Q},$$

где i – индекс обозначения оператора, которому мы поставили в соответствие номер i (это номер узла $O_i \in \mathbf{O}$ интерпретированного алгоритмом $\mathfrak{D}_i \in \mathfrak{D}$).

Пусть \mathbf{B} – расширение алфавита \mathbf{A} :

$$\mathbf{B} = \mathbf{A} \cup \bar{\mathbf{A}} \cup [\cdot]_0$$

и \mathfrak{S}_i – распространение алгоритма \mathfrak{D}_i (в алфавите \mathbf{A} для $i = 1, \dots, m$) на алфавит \mathbf{B} . Тогда искомый алгоритм \mathfrak{A}_i представляется в виде

$$\mathfrak{A}_i = \mathfrak{S}_i.$$

– если $i \in \mathbf{In}$, то

$$\mathfrak{A}_i(PU^{l-1}\bar{Q}) = \mathfrak{I}_i(P) U^{l-1}\bar{Q},$$

где i – индекс обозначения распознавателя, которому мы поставили в соответствие номер i (это номер узла $I_i \in \mathbf{I}$ интерпретированного алгоритмом $\mathfrak{I}_i \in \mathfrak{I}$).

Пусть \mathfrak{I}_i – распространение алгоритма \mathfrak{I}_i (в алфавите \mathbf{A} для $i = 1, \dots, m$) на алфавит \mathbf{B} . Тогда искомый алгоритм \mathfrak{A}_i представляется в виде

$$\mathfrak{A}_i = \mathfrak{I}_i.$$

– если $i \in \mathbf{Pn}$, то для

$$\bar{Q} = \bar{Q}_1 \bar{Q}_2^i \bar{Q}_3,$$

где \bar{Q}_1 – слово в алфавите $\bar{\mathbf{A}}^1 \cup \dots \cup \bar{\mathbf{A}}^{i-1}$, \bar{Q}_2^i – слово в алфавите $\bar{\mathbf{A}}^i$ и \bar{Q}_3 – слово в алфавите $\bar{\mathbf{A}}^{i+1} \cup \dots \cup \bar{\mathbf{A}}^p$ (вхождение слова \bar{Q}_2^i в слово \bar{Q} будем обсуждать и далее),

$$\mathfrak{A}_i(PU^{l-1}\bar{Q}) = Q_2 PU^{l-1}\bar{Q}_1 \bar{Q}_2^{P_i} \bar{Q}_3,$$

где i – индекс обозначения памяти, в которой находится k -узел с номером i (i есть номер k -узла в узле $P_i \in \mathbf{P}$); i есть определено тем самым образом и в следующих построениях.

Пусть \mathfrak{S}_i — алгоритм в алфавите $A \cup \bar{A}^1 \cup \bar{A}^2 \cup \dots \cup \bar{A}^i \cup [\cdot]_0$, для $i = 1, \dots, p$, со схемой

$$\begin{cases} \xi\eta \rightarrow \eta\xi & (\xi \in A, \eta \in [\cdot]_0 \cup \bar{A}^1 \cup \dots \cup \bar{A}^i) \\ \xi \rightarrow \bar{\xi}^i & (\xi \in A) \end{cases}$$

и Ω_i — алгоритм в алфавите $A \cup \bar{A}^1 \cup \bar{A}^2 \cup \dots \cup \bar{A}^i \cup [\cdot]_0 \cup \{\blacktriangle\}$ для $i = 1, \dots, p$, со схемой

$$\begin{cases} \xi\eta \rightarrow \eta\xi & (\xi \in \bar{A}^1 \cup \dots \cup \bar{A}^i \cup [\cdot]_0, \eta \in A) \\ \blacktriangle\xi \rightarrow \xi\bar{\xi}^i\blacktriangle & (\xi \in A) \\ \blacktriangle \rightarrow \cdot \\ \bar{\xi}^i \rightarrow \blacktriangle\bar{\xi}^i & (\xi \in A). \end{cases}$$

Тогда искомым алгоритм \mathfrak{A}_i представляется в виде

$$\mathfrak{A}_i = \Omega_i \circ \mathfrak{S}_i;$$

очевидно, что

$$\Omega_i(\mathfrak{S}_i(PU^{l-1}\bar{Q})) = \Omega_i(U^{l-1}\bar{Q}_1\bar{Q}_2^i\bar{P}^i\bar{Q}_3) = Q_2PU^{l-1}\bar{Q}_1\bar{Q}_2^i\bar{P}^iQ_3.$$

— если $i \in Pn^\wedge$, то

$$\mathfrak{A}_i(PU^{l-1}\bar{Q}) = Q_2U^{l-1}\bar{Q}.$$

Пусть Ω_i — алгоритм употребляемый в предыдущем и \mathfrak{M} — алгоритм в алфавите A со схемой

$$\{\xi \rightarrow (\xi \in A)\}.$$

Тогда искомым алгоритм \mathfrak{A}_i представляется в виде

$$\mathfrak{A}_i = \Omega_i \circ \mathfrak{M};$$

очевидно, что

$$\Omega_i(\mathfrak{M}(PU^{l-1}\bar{Q})) = \Omega_i(U^{l-1}\bar{Q}) = Q_2U^{l-1}\bar{Q}.$$

— если $i \in Pn_\wedge$, то

$$\mathfrak{A}_i(PU^{l-1}\bar{Q}) = U^{l-1}\bar{Q}_1\bar{Q}_2^i\bar{Q}_3.$$

Пусть \mathfrak{S}_i — алгоритм употребляемый в предыдущем. Тогда искомым алгоритм \mathfrak{A}_i представляется в виде

$$\mathfrak{A}_i = \mathfrak{S}_i.$$

— если $i \in Pn_\wedge^\wedge$, то

$$\mathfrak{A}_i(PU^{l-1}\bar{Q}) = U^{l-1}\bar{Q}.$$

Пусть \mathfrak{M} — алгоритм употребляемый в предыдущем. Тогда искомым алгоритм \mathfrak{A}_i представляется в виде

$$\mathfrak{A}_i = \mathfrak{M}.$$

– если $i \in \mathbf{Pno}$, то

221

$$\mathfrak{A}_i(PU^{[i]}\bar{Q}) = \begin{cases} Q_2PU_1[i]_0U_2\bar{Q}_1\bar{Q}_3, & \text{если для } U^{[i]} = U_1[i]_0^jU_2 \\ \text{есть } j = \Theta(i) + 1, \\ Q_2PU_1[i]_0^{j+1}U_2\bar{Q}_1\bar{Q}_2^j\bar{P}^i\bar{Q}_3, & \text{если для } U^{[i]} = U_1[i]_0^jU_2 \\ \text{есть } j < \Theta(i) + 1. \end{cases}$$

Пусть \mathfrak{R}_i – алгоритм в алфавите $A \cup \bar{A} \cup [\cdot]_0 \cup \{\blacktriangle\}$, для каждого $i \in \mathbf{Pno}$, со схемой

$$\begin{cases} [i]_0^{\Theta(i)} \rightarrow \blacktriangle \\ \xi\blacktriangle \rightarrow \blacktriangle \quad (\xi \in A \cup [\cdot]_0) \\ \blacktriangle\xi \rightarrow \blacktriangle \quad (\xi \in A \cup [\cdot]_0) \\ \blacktriangle \rightarrow \end{cases};$$

\mathfrak{P}_i – алгоритм в алфавите A , для $i = 1, \dots, p$, со схемой

$$\{\bar{\xi}^i \rightarrow (\xi \in A)\};$$

\mathfrak{Q}_i – алгоритм в алфавите $\{[i]_0\}$, для каждого $i \in \mathbf{Pno}$, со схемой

$$\{[i]_0^2 \rightarrow [i]_0\};$$

\mathfrak{R}_i – алгоритм в алфавите $\{[i]_0\}$, для каждого $i \in \mathbf{Pno}$, со схемой

$$\{[i]_0 \rightarrow \cdot [i]_0^2\};$$

\mathfrak{S}_i и \mathfrak{L}_i – алгоритмы употребляемые в предыдущем.

Тогда искомый алгоритм \mathfrak{A}_i представляется в виде*

$$\mathfrak{A}_i = ((\mathfrak{Q}_i \circ \mathfrak{P}_i) \vee \mathfrak{R}_i \mid \mathfrak{R}_i) \circ \mathfrak{L}_i \circ \mathfrak{S}_i;$$

очевидно, что

$$\begin{aligned} ((\mathfrak{Q}_i \circ \mathfrak{P}_i) \vee \mathfrak{R}_i \mid \mathfrak{R}_i)(\mathfrak{L}_i(\mathfrak{S}_i(PU^{[i]}\bar{Q}))) &= ((\mathfrak{Q}_i \circ \mathfrak{P}_i) \vee \mathfrak{R}_i \mid \mathfrak{R}_i)(\mathfrak{L}_i(U^{[i]}\bar{Q}_1\bar{Q}_2^j\bar{P}^i\bar{Q}_3)) = \\ &= ((\mathfrak{Q}_i \circ \mathfrak{P}_i) \vee \mathfrak{R}_i \mid \mathfrak{R}_i)(Q_2PU^{[i]}\bar{Q}_1\bar{Q}_2^j\bar{P}^i\bar{Q}_3) = \\ &= \begin{cases} \mathfrak{R}_i(\mathfrak{P}_i(Q_2PU^{[i]}\bar{Q}_1\bar{Q}_2^j\bar{P}^i\bar{Q}_3)) = \mathfrak{R}_i(Q_2PU^{[i]}\bar{Q}_1\bar{Q}_3) = Q_2PU_1[i]_0U_2\bar{Q}_1\bar{Q}_3, \\ \text{если } \mathfrak{R}_i(Q_2PU^{[i]}\bar{Q}_1\bar{Q}_2^j\bar{P}^i\bar{Q}_3) = \blacktriangle, \text{ т. е. если для} \\ U^{[i]} = U_1[i]_0^jU_2 \text{ есть } j = \Theta(i) + 1, \\ \mathfrak{R}_i(Q_2PU^{[i]}\bar{Q}_1\bar{Q}_2^j\bar{P}^i\bar{Q}_3) = Q_2PU_1[i]_0^{j+1}U_2\bar{Q}_1\bar{Q}_2^j\bar{P}^i\bar{Q}_3, \\ \text{если } \mathfrak{R}_i(Q_2PU^{[i]}\bar{Q}_1\bar{Q}_2^j\bar{P}^i\bar{Q}_3) \neq \blacktriangle, \text{ т. е. если для} \\ U^{[i]} = U_1[i]_0^jU_2 \text{ есть } j < \Theta(i) + 1. \end{cases} \end{aligned}$$

* $\mathfrak{A} \vee \mathfrak{B} \mid \mathfrak{C}$ есть обозначение для разветвления (алгоритмов) \mathfrak{A} и \mathfrak{B} , управляемого (алгоритмом) \mathfrak{C} ; подобно будет употребляться обозначение $\mathfrak{A} \mid \mathfrak{B}$ для повторения алгоритма \mathfrak{A} , управляемого алгоритмом \mathfrak{B} и обозначение $\mathfrak{A}\mathfrak{B}$ для объединения алгоритмов \mathfrak{A} и \mathfrak{B} . Эти обозначения приведены в последней работе А. А. Маркова [3].

— если $i \in \text{Pno}^\wedge \cup \text{Pno}_\wedge \cup \text{Pno}_\wedge^\wedge$, то алгоритм \mathfrak{A}_i перерабатывает слово $PU^{l_1}\bar{Q}$ во слово подобное слову $\mathfrak{A}_i(PU^{l_1}\bar{Q})$ для $i \in \text{Pno}$.

Если $i \in \text{Pno}^\wedge$, алгоритм \mathfrak{A}_i построим в виде

$$\mathfrak{A}_i = ((\mathfrak{Q}_i \circ \mathfrak{P}_i) \vee \mathfrak{R}_i | \mathfrak{R}_i) \circ \mathfrak{L}_i \circ \mathfrak{M};$$

очевидно, что

$$((\mathfrak{Q}_i \circ \mathfrak{P}_i) \vee \mathfrak{R}_i | \mathfrak{R}_i)(\mathfrak{L}_i(\mathfrak{M}(PU^{l_1}\bar{Q}))) = ((\mathfrak{Q}_i \circ \mathfrak{P}_i) \vee \mathfrak{R}_i | \mathfrak{R}_i)(\mathfrak{L}_i(U^{l_1}\bar{Q}))$$

и далее подобно как для $i \in \text{Pno}$.

Если $i \in \text{Pno}$, алгоритм \mathfrak{A}_i построим в виде

$$\mathfrak{A}_i = ((\mathfrak{Q}_i \circ \mathfrak{P}_i) \vee \mathfrak{R}_i | \mathfrak{R}_i) \circ \mathfrak{S}_i;$$

очевидно, что

$$((\mathfrak{Q}_i \circ \mathfrak{P}_i) \vee \mathfrak{R}_i | \mathfrak{R}_i)(\mathfrak{S}_i(PU^{l_1}\bar{Q})) = ((\mathfrak{Q}_i \circ \mathfrak{P}_i) \vee \mathfrak{R}_i | \mathfrak{R}_i)(U^{l_1}\bar{Q}_1\bar{Q}_2^i\bar{Q}_3)$$

и далее подобно как для $i \in \text{Pno}$.

Если $i \in \text{Pno}_\wedge^\wedge$, алгоритм \mathfrak{A}_i построим в виде

$$\mathfrak{A}_i = ((\mathfrak{Q}_i \circ \mathfrak{P}_i) \vee \mathfrak{R}_i | \mathfrak{R}_i) \circ \mathfrak{M};$$

очевидно, что

$$((\mathfrak{Q}_i \circ \mathfrak{P}_i) \vee \mathfrak{R}_i | \mathfrak{R}_i)(\mathfrak{M}(PU^{l_1}\bar{Q})) = ((\mathfrak{Q}_i \circ \mathfrak{P}_i) \vee \mathfrak{R}_i | \mathfrak{R}_i)(U^{l_1}\bar{Q})$$

и далее подобно как для $i \in \text{Pno}$.

Построим искомый алгоритм \mathfrak{F} как объединение суперпозиции $\mathfrak{U} \circ \mathfrak{F}_1$ и алгоритма \mathfrak{F}_2 :

$$\mathfrak{F} = (\mathfrak{U} \circ \mathfrak{F}_1) \tilde{\mathfrak{F}}_2,$$

где \mathfrak{U} — универсальный алгоритм над алфавитом $C \cup \{\delta\}$.

Построим алгоритм \mathfrak{F}_1 в $C \cup [.]_1 \cup \{\Delta, \delta\}$ такой, что

$$\mathfrak{F}_1(PU^{l-1}\bar{Q}\delta[1]_1\mathfrak{A}_1^n \dots [i-1]_1\mathfrak{A}_{i-1}^n \Delta [i]_1\mathfrak{A}_{i+1}^n \dots [s]_1\mathfrak{A}_s^n[s+1]_1) = PU^{l-1}\bar{Q}\delta\mathfrak{A}_i^n,$$

где $1 \leq i \leq s$; это значит, что потом

$$\mathfrak{U}(PU^{l-1}\bar{Q}\delta\mathfrak{A}_i^n) = \mathfrak{A}_i(PU^{l-1}\bar{Q}).$$

Схема этого алгоритма будет иметь вид

$$\left\{ \begin{array}{ll} \xi\Delta \rightarrow \Delta & (\xi \in C \cup [.]_1) \\ \Delta[i]_1 \rightarrow & (1 \leq i \leq s) \\ [i]_1\xi \rightarrow [i]_1 & (\xi \in C \cup [.]_1; 1 \leq i \leq s) \\ [i]_1 \rightarrow & (1 \leq i \leq s) \end{array} \right.$$

Построим алгоритм \mathfrak{B}_2 в $\mathbf{C} \cup [\cdot]_1 \cup \{\delta, \Delta\}$ такой, что

$$\begin{aligned} \mathfrak{B}_2(PU^{[1]}Q\delta[1]_1\mathfrak{A}_1^n \dots [i-1]_1\mathfrak{A}_{i-1}^n \Delta [i]_1\mathfrak{A}_i^n[i+1]_1\mathfrak{A}_{i+1}^n \dots [s]_1\mathfrak{A}_s^n[s+1]_1) = \\ = \delta[1]_1\mathfrak{A}_1^n \dots [i-1]_1\mathfrak{A}_{i-1}^n \Delta [i]_1\mathfrak{A}_i^n[i+1]_1\mathfrak{A}_{i+1}^n \dots [s]_1\mathfrak{A}_s^n[s+1]_1. \end{aligned}$$

Схема этого алгоритма имеет вид

$$\begin{cases} \xi\delta \rightarrow \delta(\xi \in \mathbf{A} \cup \bar{\mathbf{A}} \cup [\cdot]_0) \\ \delta \rightarrow \cdot\delta \end{cases}$$

2.2.2. Алгоритм \mathfrak{B}

Сопоставим каждой букве (знаку) $[i]_0$ алфавита $[\cdot]_0$ двойника обозначенного через $[i]_2$; алфавит этих двойников (т. е. двойников всех букв алфавита $[\cdot]_0$) обозначим через $[\cdot]_2$; алгоритм \mathfrak{B} построим в алфавите $\mathbf{C} \cup [\cdot]_1 \cup [\cdot]_2 \cup \{[1]_3, \dots, [s+1]_3\} \cup \{[1]_4, \dots, [s+1]_4\} \cup \{\delta, \Delta, \nabla\}$ со схемой

$$\left\{ \begin{array}{ll} \Delta [i]_1 \rightarrow [i]_2 [i]_1 & (i \in \mathbf{Pno}) \\ \xi [i]_2 \rightarrow [i]_2 \xi & (\xi \in \mathbf{C} \cup \{\delta\} \cup [\cdot]_1 \cup [\cdot]_0 \setminus \{[i]_0\}; i \in \mathbf{Pno}) \\ [i]_0 [i]_2 \rightarrow [i]_3 [\Psi(i, a-1)]_3 & (2 \leq a \leq \Theta(i)+1; i \in \mathbf{Pno}) \\ [i]_3 \xi \rightarrow \xi [i]_3 & (\xi \in \mathbf{C} \cup \{\delta\} \cup \{[1]_1, \dots, [i-1]_1\}; \\ & 1 \leq i \leq s+1) \\ [i]_3 [i]_1 \rightarrow \cdot \Delta [i]_1 & (1 \leq i \leq s+1) \\ \nabla \xi \rightarrow \xi \nabla & (\xi \in \mathbf{C} \cup \{\xi\} \cup [\cdot]_1) \\ \nabla \Delta [i]_1 \rightarrow [i]_1 \Delta [\Psi(i, 2)]_4 & (i \in \mathbf{In}) \\ \xi \Delta [i]_4 \rightarrow \Delta [i]_4 \xi & (\xi \in \mathbf{C}; 1 \leq i \leq s+1) \\ [i]_1 \Delta [i]_4 \rightarrow \cdot \Delta [i]_1 & (1 \leq i \leq s+1) \\ [i]_1 \Delta [j]_4 \rightarrow \Delta [j]_4 [i]_1 & (1 \leq i \leq s+1; 1 \leq j \leq s+1; i > j) \\ \Delta [i]_4 [j]_1 \rightarrow [j]_1 \Delta [i]_4 & (1 \leq i \leq s+1; 1 \leq j \leq s+1; i > j) \\ \Delta [i]_1 \rightarrow [i]_1 \Delta [\Psi(i, 1)]_4 & (i \in \mathbf{On} \cup \mathbf{Pn}) \\ \xi \rightarrow \xi \nabla & (\xi \in \mathbf{A}) \\ \Delta [i]_1 \rightarrow [i]_1 \Delta [\Psi(i, 1)]_4 & (i \in \mathbf{In}) \end{array} \right.$$

Можем убедиться, что если P и Q — пустые слова или слова в алфавите \mathbf{A} ; то

$$\begin{aligned} \mathfrak{B}(PU^{[1]}Q\delta[1]_1\mathfrak{A}_1^n \dots [i-1]_1\mathfrak{A}_{i-1}^n \Delta [i]_1\mathfrak{A}_i^n[i+1]_1\mathfrak{A}_{i+1}^n \dots [s]_1\mathfrak{A}_s^n[s+1]_1) = \\ = PU^{[1]}Q\delta[1]_1\mathfrak{A}_1^n \dots [\Psi(i, x)-1]_1\mathfrak{A}_{\Psi(i, x)-1}^n \Delta \\ [\Psi(i, x)]_1\mathfrak{A}_{\Psi(i, x)}^n[\Psi(i, x)+1]_1\mathfrak{A}_{\Psi(i, x)+1}^n \dots [s]_1\mathfrak{A}_s^n[s+1]_1 \end{aligned}$$

(если $i \in \mathbf{In}$ и $P = \Lambda$, то $x = 1$,

если $i \in \mathbf{In}$ и $P \neq \Lambda$, то $x = 2$,

если $i \in \mathbf{On} \cup \mathbf{Pn}$, то $x = 1$,

если $i \in \mathbf{Pno}$, то для $U^{[1]} = U_1[i]_0^g U_2$ есть $x = a-1$).

2.2.3. Алгоритм \mathfrak{G}

Алгоритм \mathfrak{G} построим в алфавите $C \cup [\cdot]_1 \cup \{\delta, \Delta\}$ со схемой

$$\begin{cases} \xi \Delta [s+1]_1 \rightarrow \Delta [s+1]_1 & (\xi \in C \cup [\cdot]_1 \cup \{\xi\}) \\ \Delta [s+1]_1 \rightarrow \end{cases};$$

если $T = PU^{l-1} Q \delta [1]_1 \mathfrak{A}_1^m \dots [i-1]_1 \mathfrak{A}_{i-1}^m \Delta [i]_1 \mathfrak{A}_i^m [i+1]_1 \mathfrak{A}_{i+1}^m \dots [s]_1 \mathfrak{A}_s^m [s+1]_1$,
то

$$\mathfrak{G}(T) = T, \text{ для } 1 \leq i \leq s;$$

$$\mathfrak{G}(PU^{l-1} Q \delta [1]_1 \mathfrak{A}_1^m \dots [s]_1 \mathfrak{A}_s^m \Delta [s+1]_1) = \Lambda.$$

2.2.4. Алгоритм \mathfrak{D}

Алгоритм \mathfrak{D} в суперпозиции \mathfrak{A} построим в алфавите $D = C \cup [\cdot]_1 \cup \{\delta, \Delta\}$ со схемой

$$\{\xi \rightarrow (\xi \in D \setminus A);$$

$$\mathfrak{D}(PU^{l-1} Q \delta [1]_1 \mathfrak{A}_1^m \dots [s]_1 \mathfrak{A}_s^m \Delta [s+1]_1) = P \quad (P - \text{слово в } A).$$

Замечание. Считаю излишним доказательство того, что — если имеет смысл выражение $\mathfrak{K}(S)$, то вышеприведенные алгоритмы нам гарантируют, что имеет смысл и выражение $\mathfrak{D}(\mathfrak{E}(\mathfrak{B}(S)))$, что для равенства

$$\mathfrak{K}(S) = T$$

имеет место равенство

$$\mathfrak{A}(S) = T,$$

где

$$\mathfrak{A} = \mathfrak{D} \circ \mathfrak{E} \circ \mathfrak{B};$$

— и наоборот, если имеет смысл выражение $\mathfrak{D}(\mathfrak{E}(\mathfrak{B}(S)))$, то способ определения алгоритмов $\mathfrak{B}, \mathfrak{E}, \mathfrak{D}$ нам гарантирует, что имеет смысл и выражение $\mathfrak{K}(S)$, что для равенства

$$\mathfrak{A}(S) = T,$$

где

$$\mathfrak{A} = \mathfrak{D} \circ \mathfrak{E} \circ \mathfrak{B},$$

имеет место равенство

$$\mathfrak{K}(S) = T.$$

(Поступило 7-го апреля 1967 г.)

- [1] Л. А. Калужнин: Об алгоритмизации математических задач. Сб. „Проблемы кибернетики“, вып. 2; Физматгиз, Москва 1959, 51—67.
- [2] А. А. Марков: Теория алгорифмов. Труды Матем. ин-та АН СССР, *XLII*. Москва—Ленинград 1954.
- [3] А. А. Марков: О некоторых алгорифмах, связанных с системами слов. Известия АН СССР, серия математическая 27 (1963), 101—160.
- [4] Н. П. Тер-Захарян: О нормализуемости граф-схемных алгорифмов. Сб. „Математические вопросы кибернетики и вычислительной техники“, Ереван 1965, 30—39.

ВЪТАН**O algoritmech s paměťovými prvky**

ZDENĚK ZASTÁVKA

V článku je definován algoritmus s paměťovými prvky pomocí grafického schématu (pojem grafického schématu je obdobou Kalužninovy definice v jeho práci [1]). Tento algoritmus je nazván kompozičním algoritmem, protože je zejména vhodný pro sestavování algoritmů kompozic daných algoritmů (kompozičních algoritmů), jejichž nejpoužívanější druhy jsou definovány ve větách 1 až 4.

Interpretujeme-li jednotlivé uzly grafického schématu definovaného algoritmu normálními algoritmy (ve smyslu práce A. A. Markova [2]), dostaneme tzv. kvazi-normální kompoziční algoritmy (k. k. a.). Ukáže se zřejmým, že ke každému normálnímu algoritmu existuje ekvivalentní k. k. a. (věta 5); je dokázáno, že ke každému k. k. a. existuje ekvivalentní normální algoritmus (věta 6.).

Dr. Zdeněk Zastávka, CSc., oddělení logiky katedry filosofie, filosofická fakulta Karlovy university, náměstí Krasnoarmějců 2, Praha 1.